性能評価モデル生成装置および性能評価モデル生成方法

APPARATUS AND METHOD FOR PRODUCING PERFORMANCE EVALUATION MODEL

発明の背景

発明の分野

本発明は、本発明は性能評価モデル生成装置および性能評価モデル生成方法に関し、特にＵＭＬ(Unified Modeling Language)モデルから性能評価モデルを生成する性能評価モデル生成装置および性能評価モデル生成方法に関する。

背景技術

製品サイクルの短期化のため、サービス分析から実現までの時間的短縮が望まれている。また、組み込み機器分野においては、情報端末の普及によって、より多機能かつ高機能なシステムが求められ、また、システムＬＳＩ(Large Scaled Integration)の登場によって、ハードウェアとソフトウェアとをより密にシステム設計する必要が生じている。

このような状況の下で、システム構成を決めて実装した後に評価を行っているようでは、時間的な短縮は望めない。そこで、システム構成を決定するに当たって、実装することなく性能などの見積もりを行う必要性が生じてくる。このニーズに対して、待ち行列理論を使った性能評価ツールが使われている。このような性能評価ツールでは、システムに入力されるジョブを待ち行列を使ってプロセッサに割り当てるという性能評価モデルを作成し、処理速度などのパラメータを与えることで性能評価を行っている。この性能評価モデルを使った性能評価は、性能評価モデルを作成するコストが高いため、強力な性能評価ができるにもかかわらず、広く使われるに至っていない。

特開平７－８４８３１号公報では、このような性能評価の支援手法として、過去に性能評価したモデル（性能評価モデル）をデータベースに保持し、以降の設計モデルの作成に役立てるようなツールが提案されている。このツールでは、性能評価をある一定期間続けることでツールに依存することなく性能評価モデルが蓄積されて、設計モデルの作成が容易になるというメリットが得られる。

しかし、上述した従来のツールによっても、システムをゼロから開発する場合で、性能評価モデルに類似のものがないときには、性能評価モデルをゼロから作成する必要があるため、性能評価モデルを作成するコストが高いという問題があった。

発明の要旨

本発明の目的は、モデル作成のための事実上の標準言語であるＵＭＬで記述されたＵＭＬモデルから性能評価モデルを自動的に生成する性能評価モデル生成装置を提供することにある。

また、本発明の他の目的は、ＵＭＬで記述されたＵＭＬモデルから性能評価モデルを自動的に生成する性能評価モデル生成方法を提供することにある。

本発明の第１の態様の性能評価モデル生成装置は、変換規則を格納する変換規則格納部と、ＵＭＬモデルを入力して解析するＵＭＬモデル解析部と、前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有する。

また、本発明の第２の態様の性能評価モデル生成装置は、利用者からの変換規則を入力する変換規則編集部と、前記変換規則編集部により入力された変換規則を格納する変換規則格納部と、ＵＭＬモデルを入力して解析するＵＭＬモデル解析部と、前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有する。

本発明の性能評価モデル生成方法は、ＵＭＬモデルの拡張記法を使い、ハードウェアリソースに割り付ける表記を定め、その表記に従ったＵＭＬモデルから性能評価モデルへの変換規則を決め、この変換規則によってＵＭＬモデルから系統的に性能評価モデルを作成する。

本発明の他の性能評価モデル生成方法は、シーケンス図の最初のメッセージをカレントメッセージにセットする工程と、シーケンス図のカレントメッセージに着目する工程と、送信先オブジェクトを定義するクラスのタイプおよび属性を調

査する工程と、送信元オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに基づいて変換規則を検索する工程と、検索された変換規則に従ってカレントメッセージを性能評価ツールのノードに変換し配置する工程と、カレントメッセージの次のメッセージがあるかどうかを判定する工程と、次のメッセージがあればこれをカレントメッセージとして制御を前記カレントメッセージに着目する工程に戻す工程とを含む。特に、前記クラスタイプおよび属性を調査する工程が、さらに、オブジェクトに着目する工程と、オブジェクトを定義するクラスに着目する工程と、オブジェクトから対応するクラスを探し該クラスに繋がるリソース関連線を探す工程と、リソース関連線が存在するかどうかを判定する工程と、リソース関連線が存在すればリソース関連のあるリソースクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程と、リソース関連線が存在しなければ元のクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程とからなる。

本発明の記録媒体は、コンピュータを、変換規則を格納する変換規則格納部，UMLモデルを入力して解析するUMLモデル解析部，および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録する。

また、本発明の記録媒体は、コンピュータを、利用者からの変換規則を入力する変換規則編集部，前記変換規則編集部により入力された変換規則を格納する変換規則格納部，UMLモデルを入力して解析するUMLモデル解析部，および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録する。

本発明の第１の効果は、システムの分析および設計で広く使われているUMLモデルを元に、性能評価モデルを作成することにより、性能評価モデルの作成が容易になり、性能評価モデルの作成にかかる時間の削減、つまりコストの削減が

できることである。

その理由は、UMLモデルで純粋に機能モデルを作成する部分に関しては、これまでのシステム設計の手法をほぼそのまま流用でき、UMLモデルを元にしてハードウェアリソースとの対応を考えるだけで性能評価モデルが生成できるためである。

第２の効果は、システムのUMLモデルに誤りが無ければ、性能評価モデルの作成時点で起こりうる性能評価モデルの作成誤りをなくすことが可能となることである。

その理由は、UMLモデルに誤りがなく、そのUMLモデルから系統的に性能評価モデルを作成するため、性能評価モデルの作成時に起こりうる単純な誤り，勘違いなどを防ぐことが可能となるからである。

第３の効果は、性能評価ツールの違いを越えて、既存のモデルや他の人が作成したモデルを広く参照することが容易になることである。

その理由は、機能モデルとしてUMLという統一モデリング言語を使うことにより、異なる性能評価ツール間でも同一のUMLモデルを使ってシステムを表現できるためである。性能評価モデルは、一般的に、独自のフォーマットでモデルを記述する必要があり、このモデルは一般的に各ツールで異なっている。本発明によれば、UMLモデルという統一表記法でシステムの機能モデルを入力し、性能評価をするため、システムの機能モデルの入力も一般的なツールが使え、性能評価ツールの違いを越えて、既存資産の流用も容易となるからである。

図面の簡単な説明

図１は、本発明の第１の実施例に係る性能評価モデル生成装置の構成を示すブロック図である。

図２は、第１の実施例に係る性能評価モデル生成方法の概要を説明する図である。

図３は、図２中のシーケンス図の内容を例示する図である。

図４は、図２中のクラス図の内容を例示する図である。

図５は、図２中の性能評価サブモデルの内容を例示する。

図６は、第１の実施例に係る性能評価モデル生成方法の手順を示すフローチャートである。

図７は、リソースクラスに対応するノード群を例示する図である。

図８は、本発明の第２の実施例に係る性能評価モデル生成装置の構成を示すブロック図である。

望ましい実施態様

以下、本発明の実施例について図面を参照して詳細に説明する。

第１の実施例

図１は、本発明の第１の実施例に係る性能評価モデル生成装置の構成を示すブロック図である。本実施例に係る性能評価モデル生成装置は、ＵＭＬモデル５を入力して解析するＵＭＬモデル解析部１と、ＵＭＬモデル解析部１による解析結果を変換規則格納部４に格納された変換規則に従って変換して性能評価モデル６を生成する変換処理部２と、利用者からの変換規則を入力する変換規則編集部３と、変換規則編集部３により入力された変換規則を格納する変換規則格納部４とから構成されている。なお、図１中、符号７はＵＭＬモデル５を生成するＵＭＬモデル入力ツール、８は性能評価モデル６を性能評価する性能評価ツールをそれぞれ示す。

図２は、システムの動作を記述したＵＭＬモデル５から性能評価モデル６を系統的に生成する性能評価モデル生成方法を説明する図である。ＵＭＬモデル５は、全てＵＭＬという統一モデリング言語の規格に従った記法で記述する。システムの動作は、動的な振る舞いを表すシーケンス図５１と、システムの静的な特徴を表すクラス図５２とで表現される。第１の実施例では、ＵＭＬモデル５の記述単位として、ＵＭＬモデル５をパッケージ５０という単位に分割する。つまり、ＵＭＬモデル５は１つ以上のパッケージ５０から構成され、パッケージ５０毎にシーケンス図５１およびクラス図５２を用意する。また、１つ

のパッケージ５０内のシーケンス図５１およびクラス図５２から生成される性能評価モデル６の単位を、性能評価サブモデル６０と呼ぶ。つまり、性能評価モデル６は、１つ以上の性能評価サブモデル６０から構成されている。

図３を参照すると、シーケンス図５１は、システムの動的な関係をモデル化した図である。シーケンス図５１の構成要素の中で、性能評価サブモデル６０の作成に関係するものには、オブジェクトＯｂ０１～Ｏｂ０６と、メッセージＭ０１～Ｍ０７とがある。オブジェクトＯｂ０１～Ｏｂ０６は、機能単位に登場するクラスのインスタンスである。具体的には、後述する図４のクラス図５２中に対応するクラスＣ０１～Ｃ０５のインスタンスである。メッセージＭ０１～Ｍ０７は、クラスＣ０１～Ｃ０５の実際の動作手順を時系列に沿って配置したものであり、上の方が相対的に早い時間を表している。破線矢印で示されたものは、手続き呼び出しの戻りを示している。戻りの破線矢印は省略してもよい。

図４は、クラス図５２およびクラス図５２に対して性能評価サブモデル６０を作成するために行われる拡張表記を説明する図である。クラス図５２は、システムを構成するクラスの静的な関連を示した図である。クラスは、モデリングに使用されるモデル要素であり、システムを構成する通常のオブジェクト指向設計で使われるクラス（以下、システムクラスという）と、ハードウェアリソースを表すクラス（以下、リソースクラスという）とに大きく区別され、それぞれさらに細かく分類される。この分類については、以下の［クラスの分類］で詳しく説明する。図４においては、クラスＣ０１～Ｃ０５がシステムクラスである。性能評価のためにクラス図５２に付加する拡張表記は、１つが、処理に必要なハードウェアリソースを表すリソースクラスＣ１１，Ｃ１２，Ｃ１３として記述する。もう１つは、これらのハードウェアリソースへの関連記述であり、これをリソース関連線Ｌ０１，Ｌ０２，Ｌ０３と名付ける。また、関連自体をリソース関連と名付け、システムクラスＣ０２，Ｃ０３，Ｃ０４をリソース関連線Ｌ０１，Ｌ０２，Ｌ０３を使ってリソースクラスＣ１１，Ｃ１２，Ｃ１３と結び付けることで、リソースクラスＣ１１，Ｃ１２，Ｃ１３のハード

ウェアリソースにシステムクラスＣ０２，Ｃ０３，Ｃ０４を割り付けるという意味づけをする。このリソース関連により、性能評価サブモデル６０の生成を可能としている。変換規則格納部４中の変換規則は、以下に示す［クラスの分類］と、［性能評価モデルの変換規則］とで構成されている。

［クラスの分類］

◎システムクラス

・タイプ<<Ａｃｔｉｖｅ>>のクラス

通常のクラス。属性およびメソッドを持ち、自立的に動作するクラス。

・タイプ<<Ｄａｔａ>>のクラス

属性が中心のクラス。メソッドは、所有する属性の読み書きのみに限定されている。

・タイプ<<Ｉｎｔｅｒｆａｃｅ>>のクラス

タイプ<<Ｉｎｔｅｒｆａｃｅ>>のクラスは、パッケージ５０内に唯一のクラスであり、外部パッケージからカレントパッケージへの全てのメッセージを受け取り、カレントパッケージ内の各クラスにメッセージを送出する。

・タイプ<<Ａｃｔｏｒ>>のクラス

タイプ<<Ａｃｔｏｒ>>のクラスは、システムの外部に存在し、外部からシステム内に何らかの働きかけを行うクラスの一種である。アクターがシステム内部にも記憶域を持つ場合、クラス図５２にタイプ<<Ａｃｔｏｒ>>のクラスとタイプ<<Ｓｔｏｒａｇｅ>>のクラスとのクラス関連を記述する関係上、アクターの記述が必要となる（後述する[変換規則]のａ．アクターを参照)。これ以外の場合、クラス図５２にアクターの記述がなくてもよい。

◎リソースクラス

リソースクラスは、ハードウェアリソースを表すクラスである。リソースクラスには、性能評価サブモデル６０に対応するノード群がある。ハードウェアリソースの占有時間など、変換したノードに設定すべきパラメータをクラスの属性に記述し、そのパラメータを使って性能評価サブモデル６０のノードに変

換する。性能評価の対象にする必要がない場合、評価対象外と記し、性能評価サブモデル６０のノードは生成しない。リソースクラスが対応する性能評価サブモデル６０のノード群の例を、図７に示す。

・タイプ<<Ｓｔｏｒａｇｅ>>のクラス

タイプ<<Ｓｔｏｒａｇｅ>>のクラスは、ハードウェアリソースの中で記憶域を定義するクラスである。記憶域の種類や性質に応じて、タイプ<<Ｓｔｏｒａｇｅ>>でさまざまなクラスを定義する。

・タイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>のクラス

タイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>のクラスは、ハードウェアリソースの中で実際の処理をするハードウェアを定義するクラスである。ソフトウェア処理の場合のＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）も、このクラスで定義する。

[性能評価モデルへの変換規則]

メッセージの送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに応じて、以下の変換規則を適用する。クラスのタイプを追加することで、この変換規則を追加することが可能となる。以下、「送信元オブジェクトを定義するクラスのタイプ　→　送信先オブジェクトを定義するクラスのタイプ」という形式で変換規則を記述する。

ａ.「タイプ<<Ａｃｔｏｒ>>　→　任意のクラスタイプ」

タイプ<<Ａｃｔｏｒ>>のクラスが最初のメッセージを送信する送信元オブジェクトになっている場合、データの発生源であるソースノードに変換する。

ｂ.「外部パッケージの任意のクラスタイプ　→　カレントパッケージのタイプ　<<Ｉｎｔｅｒｆａｃｅ>>」

カレントパッケージのタイプ<<Ｉｎｔｅｒｆａｃｅ>>のクラスに対するメッセージは、データの入り口であるエンターノードに変換する。

ｃ.「カレントパッケージのタイプ<<Ｉｎｔｅｒｆａｃｅ>>　→　外部パッケージの任意のクラスタイプ」

カレントパッケージのタイプ<<Ｉｎｔｅｒｆａｃｅ>>のクラスから外部パッケージへのメッセージは、データの出口であるリターンノードに変換する。

ｄ.「任意のクラスタイプ　→　タイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>」

タイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>のクラスが定義するリソースノードを配置する。

ｅ.「任意のクラスタイプ　→　タイプ<<Ｓｔｏｒａｇｅ>>」

タイプ<<Ｓｔｏｒａｇｅ>>のクラスが定義するリソースノードを配置する。

ｆ.「任意のクラスタイプ　→　外部パッケージのタイプ<<Ｉｎｔｅｒｆａｃｅ>>」

外部パッケージのタイプ<<Ｉｎｔｅｒｆａｃｅ>>のクラスがメッセージを受け取る場合、外部パッケージに対する処理の依頼であるので、その外部パッケージに対応する性能評価サブモデル６０に変換する。これに続くメッセージが、カレントメッセージと逆方向になっている場合、これに続くメッセージを無視する。

ｇ．破線矢印

破線矢印のメッセージは、それ自体省略しても可能なものである。よって、このメッセージは、変換の際に無視することができる。

ｈ．終了処理

最後のメッセージでシンクノードまたはリターンノードが配置されない場合、このどちらかのノードを配置し、性能評価サブモデル６０を完結させる。どちらのノードを配置するかは、変換した性能評価サブモデル６０の最初のノードに着目し、シンクノードの場合はソースノード、エンターノードの場合はリターンノードを配置する。

図５を参照すると、性能評価サブモデル６０は、システム構成をノードで表現する。ノードは、大別すると、システムの処理フロー自体を表すノードと、処理フローで利用されるハードウェアリソースを表すノードとに区別できる。図５中の性能評価サブモデル６０では、前者が下側に、後者が上側に記述されている。システムの処理フローを表現するノードは、そのフローをノード間の

結線で表現している。このようなノードには、ハードウェアを消費する「処理」を表すサービスノード，データの入力を表すソースノード，処理の終了を表すシンクノードなどがある。一方、ハードウェアリソースを表すノードとして、ハードウェアに相当するリソースノードがある。リソースノードは、サービスノードでの処理によって消費される。この消費時間，優先順位などを動的パラメータ６１として設定する。さらに、入力されるデータのパラメータを加えて、性能評価サブモデル６０が生成できる。

図５は、性能評価サブモデル６０の一例を記述する図である。なお、ここでは、性能評価ツール８として、待ち行列理論を使ったＳＥＳ／Ｗｏｒｋｂｅｎｃｈ（ＳＥＳ社の登録商標）を想定している。この性能評価サブモデル６０には、処理する側のノードとして、上部に、ＣＰＵを表すサービスノードＮ０１が記述されている。一方、処理フローを表すノードとして、下部に、ＣＰＵを消費するサービスノードＮ０３と、ＣＰＵとは無関係に時間を消費するサービスノードＮ０５と、システムに入力されるデータに関連するソースノードＮ０２およびシンクノードＮ０８と、その他のノードとして、性能評価サブモデル６０への参照を表すサブモデルノードＮ０７，リソース管理のアロケートノードＮ０４，リリースノードＮ０６と、そして、その間の結線３１１～３１６が記述されている。最後に、上部と下部とを結び付ける記述として、動的パラメータ６１が設定されている。変換では、リソース管理のアロケートノードＮ０４，メモリアクセスを表すサービスノードＮ０５，およびリリースノードＮ０６という一連のノードを排他アクセスＭｅｍｏｒｙのモデルとしてあらかじめ用意しておき、これらをセットにして変換規則を適用する。図５にあるノード以外に、他の性能評価サブモデル６０からデータの流れを引き継ぐエンターノードおよびリターンノードがある。

[性能評価サブモデルへの変換手順]

図６は、性能評価サブモデル６０への変換手順を表すフローチャートである。性能評価サブモデル６０は、あるパッケージ５０に着目し、そのパッケージ５

０のシーケンス図５１およびクラス図５２から変換して生成される。この着目しているパッケージ５０をカレントパッケージと呼び、それ以外のパッケージ５０を外部パッケージと呼ぶ（図４参照）。

図６を参照すると、変換処理部２の処理は、カレントメッセージセットステップＳ１０１と、カレントメッセージ着目ステップＳ１０２と、送信元オブジェクトのクラスタイプおよび属性調査ステップＳ１０３と、送信先オブジェクトのクラスタイプおよび属性調査ステップＳ１０４と、変換規則検索ステップＳ１０５と、ノード生成・連結ステップＳ１０６と、次メッセージ有無判定ステップＳ１０７と、オブジェクト着目ステップＳ１０８と、クラス着目ステップＳ１０９と、リソース関連探索ステップＳ１１０と、リソース関連有無判定ステップＳ１１１と、リソース関連先クラスタイプおよび属性保存ステップＳ１１２と、自クラスタイプおよび属性保存ステップＳ１１３とからなる。

次に、このように構成された第１の実施例に係る性能評価モデル生成装置の動作について、第１の実施例に係る性能評価モデル生成方法とともに説明する。

ここでは、図３のシーケンス図５１および図４のクラス図５２からメッセージを時系列に沿って図５の性能評価サブモデル６０に変換する例について説明する。利用者は、変換規則編集部３を使用して、あらかじめ変換規則を編集して変換規則格納部４に格納しておく。ここでは、既述した変換規則ａ．～ｈ．が格納されたものとする。

一方、ＵＭＬモデル解析部１は、ＵＭＬモデル入力ツール７を使って生成されたＵＭＬモデル５を入力して解析する。詳しくは、性能評価サブモデル６０に変換するために不必要な情報を削除し、必要な情報が揃っているかどうかをチェックし、必要な情報が揃っていたならば変換処理部２に解析結果を渡す。

変換処理部２は、ＵＭＬモデル解析部１から渡された解析結果中の、シーケンス図５１の最初のメッセージＭ０１をカレントメッセージにセットする（ステップＳ１０１）。

次に、変換処理部２は、シーケンス図５１のカレントメッセージＭ０１に着目する（ステップＳ１０２）。

続いて、変換処理部２は、カレントメッセージＭ０１の送信元オブジェクトＯｂ０１を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０３）。

詳しくは、変換処理部２は、まず、送信元オブジェクトＯｂ０１に着目し（ステップＳ１０８）、名前「ａｃｔｏｒ」に基づいて送信元オブジェクトＯｂ０１を定義するクラス（ここでは、アクターはシステム内部に記憶域を持たないものとし、図４中には対応するクラスが図示されていない）に着目する（ステップＳ１０９）。次に、そのクラスに繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線が存在しないので（ステップＳ１１１）、元のクラスのタイプ<<Ａｃｔｏｒ>>および属性をタイプおよび属性としてそのまま保存する（ステップＳ１１３）。

次に、変換処理部２は、カレントメッセージＭ０１の送信先オブジェクトＯｂ０２を定義するクラスＣ０１のタイプおよび属性を調査する（ステップＳ１０４）。

詳しくは、変換処理部２は、まず、送信先オブジェクトＯｂ０２に着目し（ステップＳ１０８）、名前「ＭａｉｎＰ」に基づいて送信先オブジェクトＯｂ０２を定義するシステムクラスＣ０１に着目する（ステップＳ１０９）。次に、システムクラスＣ０１に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線が存在しないので（ステップＳ１１１）、元のシステムクラスＣ０１のタイプ<<Ｉｎｔｅｒｆａｃｅ>>および属性をシステムクラスＣ０１のタイプおよび属性としてそのまま保存する（ステップＳ１１３）。

続いて、変換処理部２は、送信元オブジェクトＯｂ０１を定義するクラスのタイプ<<Ａｃｔｏｒ>>および送信先オブジェクトＯｂ０２を定義するシステムクラスＣ０１のタイプ<<Ｉｎｔｅｒｆａｃｅ>>に基づいて、変換規則格納部４から変換規則ａ．を検索する（ステップＳ１０５）。

続いて、変換処理部２は、検索された変換規則ａ．「アクターが最初のメッセージを送信するオブジェクトになっている場合、データの発生源であるソースノードに変換する。」に基づいて、ソースノードＮ０２に変換して性能評価サブ

モデル６０に配置する（ステップＳ１０６）。

次に、変換処理部２は、シーケンス図５１の次のメッセージの存在を調査し（ステップＳ１０７）、次のメッセージＭ０２があるので、ステップＳ１０２に制御を戻す。変換処理部２は、次のメッセージＭ０２をカレントメッセージにセットして着目する（ステップＳ１０２）。

次に、変換処理部２は、メッセージＭ０２の送信元オブジェクトＯｂ０２を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０３）。詳しくは、変換処理部２は、まず、送信元オブジェクトＯｂ０２に着目し（ステップＳ１０８）、名前「ＭａｉｎＰ」に基づいて送信元オブジェクトＯｂ０２を定義するシステムクラスＣ０１に着目する（ステップＳ１０９）。次に、システムクラスＣ０１に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線が存在しないので（ステップＳ１１１）、元のシステムクラスＣ０１のタイプ<<Ｉｎｔｅｒｆａｃｅ>>および属性をシステムクラスＣ０１のタイプおよび属性としてそのまま保存する（ステップＳ１１３）。

続いて、変換処理部２は、メッセージＭ０２の送信先オブジェクトＯｂ０３を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０４）。詳しくは、変換処理部２は、まず、送信先オブジェクトＯｂ０３に着目し（ステップＳ１０８）、名前「Ｍａｉｎ」に基づいて送信先オブジェクトＯｂ０３を定義するシステムクラスＣ０２に着目する（ステップＳ１０９）。次に、システムクラスＣ０２に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線Ｌ０１が存在するので（ステップＳ１１１）、リソース関連のあるリソースクラスＣ１１のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および属性「評価対象」をシステムクラスＣ０２のタイプとして保存する（ステップＳ１１２）。

次に、変換処理部２は、送信元オブジェクトＯｂ０２を定義するシステムクラスＣ０１のタイプ<<Ｉｎｔｅｒｆａｃｅ>>および送信先オブジェクトＯｂ０３を定義するシステムクラスＣ０２のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>に基づいて、変換規則格納部４から変換規則ｄ．を検索する（ステップＳ１０５）。

続いて、変換処理部２は、変換規則ｄ．「タイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>

のクラスが定義するリソースノードを配置する。」を適用して、図７に示すように、タイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>のリソースクラスＣ１１が定義するリソースノードＮ０１に変換して配置する（ステップＳ１０６）。

次に、変換処理部２は、シーケンス図５１の次のメッセージの存在を調査（ステップＳ１０７）、次のメッセージＭ０３があるので、ステップＳ１０２に制御を戻す。

続いて、変換処理部２は、次のメッセージＭ０３をカレントメッセージにセットして着目する（ステップＳ１０２）。

次に、変換処理部２は、メッセージＭ０３の送信元オブジェクトＯｂ０３を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０３）。

詳しくは、変換処理部２は、まず、送信元オブジェクトＯｂ０３に着目し（ステップＳ１０８）、名前「Ｍａｉｎ」に基づいて送信元オブジェクトＯｂ０３を定義するシステムクラスＣ０２に着目する（ステップＳ１０９）。次に、システムクラスＣ０２に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線Ｌ０１が存在するので（ステップＳ１１１）、リソース関連のあるリソースクラスＣ１１のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および属性「評価対象」をシステムクラスＣ０２のタイプおよび属性として保存する（ステップＳ１１２）。

続いて、変換処理部２は、メッセージＭ０３の送信先オブジェクトＯｂ０４を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０４）。詳しくは、変換処理部２は、まず、送信先オブジェクトＯｂ０４に着目し（ステップＳ１０８）、名前「Ａ」に基づいて送信先オブジェクトＯｂ０４を定義するシステムクラスＣ０３に着目する（ステップＳ１０９）。次に、システムクラスＣ０３に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線Ｌ０２が存在するので（ステップＳ１１１）、リソース関連のあるリソースクラスＣ１２のタイプ<<Ｓｔｏｒａｇｅ>>および属性「評価対象外」をシステムクラスＣ０３のタイプおよび属性として保存する（ステップＳ１１２）。

次に、変換処理部２は、送信元オブジェクトＯｂ０３を定義するシステムクラスＣ０２のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および送信先オブジェクトＯｂ

０４を定義するシステムクラスＣ０３のタイプ<<Ｓｔｏｒａｇｅ>>に基づいて、変換規則格納部４から変換規則ｅ．を検索する（ステップＳ１０５）。

続いて、変換処理部２は、変換規則ｅ．「タイプ<<Ｓｔｏｒａｇｅ>>のクラスが定義するリソースノードを配置する。」を適用して、図７に示すように、タイプ<<Ｓｔｏｒａｇｅ>>のクラスＣ１２が定義するリソースノード群Ｎ０４，Ｎ０５，Ｎ０６に変換し、リソースクラスＣ１２の属性が「評価対象」となっているので、ノード群Ｎ０４，Ｎ０５，Ｎ０６を評価対象サブモデル６０に配置する（ステップＳ１０６）。

次に、変換処理部２は、シーケンス図５１の次のメッセージの存在を調査し（ステップＳ１０７）、次のメッセージＭ０４があるが、破線矢印のメッセージであるので、これを無視し、さらに次のメッセージの存在を調査し、メッセージＭ０５があるので、ステップＳ１０２に制御を戻す。変換処理部２は、次のメッセージＭ０５をカレントメッセージにセットして着目する（ステップＳ１０２）。

次に、変換処理部２は、カレントメッセージＭ０５の送信元オブジェクトＯｂ０３を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０３）。

詳しくは、変換処理部２は、まず、送信元オブジェクトＯｂ０３に着目し（ステップＳ１０８）、名前「Ｍａｉｎ」に基づいて送信元オブジェクトＯｂ０３を定義するシステムクラスＣ０２に着目する（ステップＳ１０９）。次に、システムクラスＣ０２に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線Ｌ０１が存在するので（ステップＳ１１１）、リソース関連のあるリソースクラスＣ１１のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および属性「評価対象」をシステムクラスＣ０２のタイプおよび属性として保存する（ステップＳ１１２）。

続いて、変換処理部２は、メッセージＭ０５の送信先オブジェクトＯｂ０５を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０４）。詳しくは、変換処理部２は、まず、送信先オブジェクトＯｂ０５に着目し（ステップＳ１０８）、名前「Ｂ」に基づいて送信先オブジェクトＯｂ０５を定義す

るシステムクラスＣ０４に着目する（ステップＳ１０９）。次に、システムクラスＣ０４に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線Ｌ０３が存在するので（ステップＳ１１１）、リソースクラスＣ１３のタイプ<<Ｓｔｏｒａｇｅ>>および属性「評価対象外」をシステムクラスＣ０４のタイプおよび属性として保存する（ステップＳ１１２）。

次に、変換処理部２は、送信元オブジェクトＯｂ０３を定義するシステムクラスＣ０２のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および送信先オブジェクトＯｂ０５を定義するシステムクラスＣ０４のタイプ<<Ｓｔｏｒａｇｅ>>に基づいて、変換規則格納部４から変換規則ｅ．を検索する（ステップＳ１０５）。

続いて、変換処理部２は、変換規則ｅ．を適用しようとするが、システムクラスＣ０４の属性が「評価対象外」となっているので、ノードへの変換を行わずに性能評価サブモデル６０に配置しない（ステップＳ１０６）。

次に、変換処理部２は、シーケンス図５１の次のメッセージの存在を調査し（ステップＳ１０７）、次のメッセージＭ０６があるので、ステップＳ１０２に制御を戻す。

次に、変換処理部２は、シーケンス図５１のカレントメッセージＭ０６に着目する（ステップＳ１０２）。

続いて、変換処理部２は、メッセージＭ０６の送信元オブジェクトＯｂ０３を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０３）。

詳しくは、変換処理部２は、まず、送信元オブジェクトＯｂ０３に着目し（ステップＳ１０８）、名前「Ｍａｉｎ」に基づいて送信元オブジェクトＯｂ０３を定義するシステムクラスＣ０２に着目する（ステップＳ１０９）。次に、システムクラスＣ０２に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線Ｌ０１が存在するので（ステップＳ１１１）、リソース関連のあるリソースクラスＣ１１のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および属性「評価対象」をシステムクラスＣ０２のタイプおよび属性として保存する（ステップＳ１１２）。

次に、変換処理部２は、メッセージＭ０６の送信先オブジェクトＯｂ０６を定義するシステムクラスのタイプおよび属性を調査する（ステップＳ１０４）。

詳しくは、変換処理部２は、まず、送信先オブジェクトＯｂ０６に着目し（ステップＳ１０８）、名前「ｓｕｂＰ」に基づいて送信先オブジェクトＯｂ０６を定義するシステムクラスＣ０５に着目する（ステップＳ１０９）。次に、システムクラスＣ０５に繋がるリソース関連線を探す（ステップＳ１１０）。リソース関連線が存在しないので（ステップＳ１１１）、元のシステムクラスＣ０５のタイプ<<Ｉｎｔｅｒｆａｃｅ>>および属性をシステムクラスＣ０５のタイプおよび属性として保存する（ステップＳ１１３）。

次に、変換処理部２は、送信元オブジェクトＯｂ０３を定義するシステムクラスＣ０３のタイプ<<Ｐｒｏｃｅｓｓｉｎｇ>>および送信先オブジェクトＯｂ０６を定義するシステムクラスＣ０５のタイプ<<Ｉｎｔｅｒｆａｃｅ>>に基づいて、変換規則格納部４から変換規則ｆ．を検索する（ステップＳ１０５）。

続いて、変換処理部２は、変換規則ｆ．を適用して、外部パッケージのタイプ<<Ｉｎｔｅｒｆａｃｅ>>のクラスがメッセージを受け取る場合、外部パッケージに対する処理の依頼であるので、その外部パッケージに対応する性能評価サブモデル６０のノードＮ０７に変換し配置する（ステップＳ１０６）。このノードＮ０７は、タイプ<<Ｉｎｔｅｒｆａｃｅ>>のシステムクラスＣ０５が属するパッケージ５０を表している。

次に、変換処理部２は、シーケンス図５１の次のメッセージの存在を調査し、次のメッセージＭ０７があるが、破線矢印のメッセージであるので、これを無視し、さらに次のメッセージの存在を調査し、メッセージがないと判断する（ステップＳ１０７）。

最後に、変換処理部２は、性能評価サブモデル６０がシンクノードまたはリターンノードで完結していないので、変換規則ｈ．により、性能評価サブモデル６０に終了処理を加える。この性能評価サブモデル６０では、最初のオブジェクトＯｂ０１をソースノードＮ０２としたので、シンクノードＮ０８を配置し、性能評価サブモデル６０を完結させる。

なお、第１の実施例では、変換規則ａ．～ｈ．が変換規則格納部４に格納されているものとしたが、変換規則は変換規則編集部３により変換規則格納部４

に追加および変更可能となっているので、上記に示した変換規則以外に従った変換も可能である。

また、第１の実施例では、１つのパッケージ５０の性能評価サブモデル６０について説明したが、この処理を繰り返すことにより複数の性能評価サブモデル６０からなる性能評価モデル６の全体が同様にして得られる。

第１の実施例では、通常のシステム分析およびシステム設計で行われるような機能的な観点からＵＭＬモデル５を作成し、ハードウェアリソースを表すリソースクラスと、システムクラスとリソースクラスとの関連を示すリソース関連線とを追加することで、性能評価モデル６を生成することが可能となる。これにより、ＵＭＬモデル５の作成と性能評価モデル６の作成との距離が縮まり、最終的に、性能評価モデル６の作成時間および作成コストを軽減することが可能となる。

第２の実施例

図８を参照すると、本発明の第２の実施例に係る性能評価モデル生成装置は、図１に示した第１の実施例に係る性能評価モデル生成装置に対して、性能評価モデル生成プログラムを記録した記録媒体１００を備える点のみが異なっている。この記録媒体１００は、磁気ディスク，半導体メモリ，その他の記録媒体であってよい。

性能評価モデル生成プログラムは、記録媒体１００からコンピュータでなる性能評価モデル生成装置に読み込まれ、ＵＭＬモデル解析部１，変換処理部２，変換規則編集部３，および変換規則格納部４として動作する。各部１～４の動作は、第１の実施例に係る性能評価モデル生成装置における対応する各部１～４と全く同様になるので、その詳しい説明は省略する。

特許請求の範囲

１．　変換規則を格納する変換規則格納部と、
ＵＭＬモデルを入力して解析するＵＭＬモデル解析部と、
前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部と
を有する性能評価モデル生成装置。

２．　利用者からの変換規則を入力する変換規則編集部と、
前記変換規則編集部により入力された変換規則を格納する変換規則格納部と、
ＵＭＬモデルを入力して解析するＵＭＬモデル解析部と、
前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部と
を有する性能評価モデル生成装置。

３．　ＵＭＬモデルの拡張記法を使い、ハードウェアリソースに割り付ける表記を定め、その表記に従ったＵＭＬモデルから性能評価モデルへの変換規則を決め、この変換規則によってＵＭＬモデルから系統的に性能評価モデルを作成する性能評価モデル生成方法。

４．　シーケンス図の最初のメッセージをカレントメッセージにセットする工程と、
シーケンス図のカレントメッセージに着目する工程と、
送信先オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、
送信元オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、
送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに基づいて変換規則を検索する工程と、

検索された変換規則に従ってカレントメッセージを性能評価ツールのノードに変換し配置する工程と、

カレントメッセージの次のメッセージがあるかどうかを判定する工程と、

次のメッセージがあればこれをカレントメッセージとして制御を前記カレントメッセージに着目する工程に戻す工程と

を含む性能評価モデル生成方法。

5．　前記クラスタイプおよび属性を調査する工程が、さらに、オブジェクトに着目する工程と、オブジェクトを定義するクラスに着目する工程と、オブジェクトから対応するクラスを探し該クラスに繋がるリソース関連線を探す工程と、リソース関連線が存在するかどうかを判定する工程と、リソース関連線が存在すればリソース関連のあるリソースクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程と、リソース関連線が存在しなければ元のクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程とからなる請求項４記載の性能評価モデル生成方法。

6．　コンピュータを、変換規則を格納する変換規則格納部，ＵＭＬモデルを入力して解析するＵＭＬモデル解析部，および前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録した記録媒体。

7．　コンピュータを、利用者からの変換規則を入力する変換規則編集部，前記変換規則編集部により入力された変換規則を格納する変換規則格納部，ＵＭＬモデルを入力して解析するＵＭＬモデル解析部，および前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録した記録媒体。

要　約　書

本発明の性能評価モデル生成装置は、変換規則を格納する変換規則格納部と、ＵＭＬモデルを入力して解析するＵＭＬモデル解析部と、前記ＵＭＬモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有する。